First Hit

☐ Generate Collection

L3: Entry 4 of 54 File: PGPB Sep 26, 2002

DOCUMENT-IDENTIFIER: US 20020138640 A1
TITLE: Apparatus and method for improving the delivery of software applications and associated data in web-based systems

Abstract Paragraph:
An improved system for streaming a software application to a plurality of clients comprises a principal server having the software stored thereon as a plurality of blocks and a plurality of intermediate servers between the principal server and the clients. The principal server is configured to stream program and data blocks to downstream devices in accordance with a dynamic prediction of the needs of those devices. The intermediate servers are configured to cache blocks received from connected upstream devices and service requests for blocks issued from downstream devices. In addition, the intermediate servers are further configured to autonomously predict the needs of downstream devices, stream the predicted blocks to the downstream devices, and if the predicted blocks are not present in the intermediate server cache, request those blocks from upstream devices. The intermediate servers can also be configured to make intelligent cache purging decisions with reference to the contents of the caches in other connected devices.

Application Filing Date:
20001222

Summary of Invention Paragraph:
[0003] The present invention relates generally to improving the delivery of software applications and associated data in web-based systems, and more particularly, to a multi-level intelligent caching system customized for use in application streaming environments.

Summary of Invention Paragraph:
[0015] The present invention relates generally to a method and system for improving the delivery of software applications and associated data, which can be stored in databases, via a network, such as the Internet. One or more intermediate tiers of intelligent caching servers are placed between a principal application server and the streaming application clients. The intermediate servers store streamed blocks, such as software application modules or streamlets and other database modules, as they are transmitted from the principal server to a client. As a result, further requests by the same client or other clients associated with the intermediate servers for previously stored information can be streamed from the intermediate servers without accessing the principal server.

Detail Description Paragraph:
[0025] FIG. 1 is a high level diagram of a system 10 for streaming software applications to one or more clients. The system comprises a streaming application server 110 which contains the application to be streamed and associated data. The application to be streamed is broken into discrete parts, segments, modules, etc., also referred to generally herein as blocks. The streams of blocks 18 can be sent individually to a client. Preferably, a predictive model of when the various blocks in the application are used as the program executes is provided at the server and this model issued to select the most appropriate block to send to a client at a given point during the execution of the application on that client's system.

Various predictive models can be used, such as a predictive tree, a neural network, a statistical database, or other probabilistic or predictive modeling techniques known to those of skill in the art. Particular predictive techniques are discussed below.

Detail Description Paragraph:
[0028] Turning to FIG. 2, there is shown a multi-level caching server system 210 configured for use in conjunction with a principal streaming application server 110. The intermediate servers 210 are situated between the principal server 110 and the various clients 220-240. One or more intermediate servers can be provided to establish multiple paths from the clients to the server 110. In a preferred embodiment, the intermediate servers 210 are arranged in a tree configuration with multiple levels or tiers as shown. The physical location of the intermediate servers 210 which make up the several tiers can vary. For example, a higher level tier server 180 can cover a specific geographic region, while lower level tier servers 190, 200 can cover specific states within that region.

CLAIMS:

18. The system of claim 16, wherein devices connected to the server are organized in a tree configuration and the communication streaming manager is configured to broadcast caching and purging event indications to direct descendant and direct ancestor devices connected to the server.

First Hit Fwd Refs

☐ Generate Collection

L3: Entry 13 of 54 File: USPT Mar 25, 2003

DOCUMENT-IDENTIFIER: US 6539382 B1
TITLE: Intelligent pre-caching algorithm for a directory server based on user data access history

Application Filing Date (1):
19990429

Brief Summary Text (7):
To this end, the Lightweight Directory Access Protocol (LDAP) has emerged as an IETF open standard to provide directory services to applications ranging from e-mail systems to distributed system management tools. LDAP is an evolving protocol that is based on a client-server model in which a client makes a TCP/IP connection to an LDAP server, sends requests, and receives responses. The LDAP information model in particular is based on an "entry," which contains information about some object. Entries are typically organized in a specified tree structure, and each entry is composed of attributes.

Drawing Description Text (8):
FIG. 6 is a representative LDAP directory tree;

Detailed Description Text (3):
The directory tree is organized in a predetermined manner, with each entry uniquely named relative to its sibling entries by a "relative distinguished name" (RDN). An RDN comprises at least one distinguished attribute value from the entry and, at most, one value from each attribute is used in the RDN. According to the protocol, a globally unique name for an entry, referred to as a "distinguished name" (DN), comprises a concatenation of the RDN sequence from a given entry to the tree root.

Detailed Description Text (4):
The LDAP search can be applied to a single entry (a base level search), an entry's children (a one level search), or an entire subtree (a subtree search). Thus, the scope supported by LDAP search are: base, one level and subtree. LDAP does not support search for arbitrary tree levels and path enumeration.

Detailed Description Text (13):
By way of brief background, and with reference to the representative directory tree in FIG. 6, a typical LDAP search may be as follows: ldapsearch -b "c=IBM_US, c=US" -s sub (sn=smith); where: "-b" is the base distinguished name (DN) from which the search is initiated, "-s" is the scope of the search (either base, one or subtree), and "sn=smith" is the filter string that specifies the condition the search needs to find matches in the directory. The three elements above form a unique combination of criteria for a given LDAP search query. As is well known, a filter key is a combination of filter string+scope+base DN, and each filter key is thus unique for an LDAP search. In the above example, the filter key is: ##EQU1## where 2 is a numeric representation of the scope subtree level (with base level being 0 and one level being 1).

Detailed Description Text (34):
One of ordinary skill will appreciate that the invention can be applied to any relational database management system (RDBMS). In addition, the EID sets approach

can also be applied to b-tree based LDAP server implementation. The inventive algorithm may be applied to any LDAP server implementation, including Netscape Directory Server.

First Hit Fwd Refs

☐ Generate Collection

L18: Entry 1 of 2 File: USPT Feb 3, 2004

DOCUMENT-IDENTIFIER: US 6687817 B1
TITLE: Configuration of a network device via the network

Application Filing Date (1):
20001114

CLAIMS:

1. A method of configuring a network device comprising: sending a configuration request from a first device to a second device via multicast; generating configuration data for the first device on the second device; sending configuration data from the second device to the first device via multicast; and writing configuration data into one or more files used by an operating system for network configuration on the first device.

16. A method of configuring a network device comprising: sending a configuration request on a predetermined multicast port of the network device; responsive to receiving a configuration pending message on the predetermined multicast port, listening for a packet containing configuration data; and writing the configuration data into one or more files used by an operating system of the network device for configuring network settings.

40. A machine-readable medium having stored thereon data representing sequences of instructions, said sequences of instructions which, when executed by a processor, cause said processor to configure a network device by: repeatedly sending a configuration request on a predetermined multicast port of the network device; responsive to receiving a configuration pending message on the predetermined multicast port, listening for a packet containing configuration data; and writing the configuration data into one or more files used by an operating system of the network device for configuring network settings.

First Hit Fwd Refs

☐ Generate Collection

L18: Entry 2 of 2 File: USPT Feb 4, 2003

DOCUMENT-IDENTIFIER: US 6515967 B1
TITLE: Method and apparatus for detecting a fault in a multicast routing infrastructure

Application Filing Date (1):
19980630

Brief Summary Text (17):
In another aspect of the invention, a format of a test sender request message for configuring a network device to transmit test data packets using a multicast routing protocol is described. In a preferred embodiment, a test sender request message includes an originator identifier field, a target identifier field, and a test group identifier field. The originator identifier field contains an identifier, such as an Internet Protocol address, corresponding to a multicast routing manager device. The target identifier field contains an identifier corresponding to a test sender device. The test group identifier field contains an identifier corresponding to a group of test receivers that includes one or more test receivers. In another preferred embodiment, the test sender request message includes a packet delay field used in the emission of data packets from the sender data packet routing device.

Brief Summary Text (18):
In another aspect of the invention, a format of a test receiver request message for configuring a network device to receive test data packets using a multicast routing protocol is described. In a preferred embodiment, a test receiver request message includes an originator identifier field, a test group identifier field, and a test sender identifier field. The originator identifier field contains an identifier, such as an Internet Protocol address, corresponding to a multicast routing manager device. The test group identifier field contains an identifier corresponding to a group of test receivers. The test sender identifier field contains an identifier corresponding to a test sender that will be sending the test receiver test data packets. In another preferred embodiment, the format of a test receiver request message includes one or more fault data transmission fields containing data relating to when fault data should be transmitted to the multicast routing manager device.

CLAIMS:

1. In a test monitoring device, a method of detecting a fault in a multicast routing infrastructure, the method comprising: sending a source configuration request for configuring a device in the infrastructure to be a test packet source, the source configuration request containing an identifier identifying the test monitoring device; sending a receiver configuration request for configuring one or more devices in the infrastructure to be test packet receivers where the test packet source sends out test packets to a predetermined group of test packet receivers, the predetermined group having a group identifier; and examining data reports from the test packet receivers relating to a plurality of test packets sent by the test packet source where the data reports are examined shortly after the data reports are provided by the test packet receivers.

First Hit Fwd Refs

☐ Generate Collection

L21: Entry 3 of 11 File: USPT Aug 27, 2002

DOCUMENT-IDENTIFIER: US 6442584 B1
TITLE: Methods for resource consolidation in a computing environment

Application Filing Date (1):
19980515

Brief Summary Text (5):
Computers are very powerful tools for storing and providing access to vast amounts of information. For instance, computer database servers are a common mechanism for storing information on computer systems while providing easy access to users. To support multiple clients concurrently accessing the system, the system must employ some sort of mechanism for managing its resources--that is, the devices, software, services, and subsystems which comprise the system. One approach is to provide resource management groups which are hierarchically formed. For instance, each resource management group may include a resource management process for managing resources allocated to its group, at least one process which is a descendant of the resource management process and not included in other resource management groups, and a resource management block for storing information on the resources managed by its own group. The resource management processes can be linked to each other in accordance with parent-child relationships between the respective resource management groups to form a resource management tree.

Detailed Description Text (16):
Complicating the whole procedure further is the fact that the edges between two nodes are one of two types, use or contain. A contain relationship means that if the node is copied, than the descendant tree must also be copied. A use relationship means that even if the node is copied, it may still be possible to share the descendants with the Extend method. For example, once b is determined not to fit into the primary, g and h will need to be copied (because they are contained), but k is still a candidate for unification (because it is used). If in fact a had already been mapped to A, then k will already be mapped to H, and therefore when b's subgraph is added, it will already have a constraint on it.

Detailed Description Text (21):
The methodology may be implemented as outlined below. Each method (with its routine names presented in parentheses) is followed by exemplary method steps. Resource Consolidation (consolidate): For each resource set: Invoke Minimal Extension on the set Minimal Extension (extend): Initialize the nodes in the resource set For each resource in the set, iterating through in tcf_hierarchy order: Invoke Extend on the resource Clean up Extend (extend resource) Try to Unify the resource, record the result in mapped_res (i.e., mapped resource) If Unify failed, copy the subgraph rooted at the resource with Add, saving the result in mapped_res Return mapped_res Unify (unify_resource) If resource is already bound, return binding For each primary resource in the same class as the secondary resource: Remove all the virtual bindings (this_merge_list, reset_merge_list) Call Compare to compare the primary and secondary resources If they match, call Bind to bind the two resources together, and return the primary resource If no primary resource matches, return a failure Compare (compare) If the primary resource is already bound: Set map to the binding If map equals the secondary resource, return success, since the two resources are already bound together Otherwise, return failure, since the primary

resource is bound to something else If the primary resource is not yet, bound: If the secondary resource is bound, then return failure, since it must be bound to something other than the primary resource Repeat the above steps by checking for bindings on the virtual list If neither the primary nor secondary resources are yet bound, in either the real or the virtual list, invoke Compare 1 to actually compare the two resources Compare1(compare1) If the comparison between the primary and secondary resources has already taken place, then get the result from the cache Otherwise, do the node comparison, and store the result in the cache; the node comparison assumes that the two nodes being compared are in the same class If the two nodes are not compatible, return failure Otherwise, for each type of edge label descending from the primary node: Invoke Compare on the children of the primary and secondary resource with the specified label If the Compare fails, return failure The primary resource and secondary resource are compatible, as are their descendants. Create a virtual binding between the two resources. Bind (do_merge) For each binding in the virtual bindings list: Do any resource-specific merging of the two resources Call BindResource to bind the two resources to each other Add (add_resource_tree) Create a new node that is an exact copy of the node in the secondary graph Create a real binding between the original, secondary resource and the new, primary resource For each contained edge label of the original resource's class: For each child connected to the original resource through the specified label: Invoke Add to add the child, recording the name of the resulting copy Change the links in the copy to point to the copies of the children For each used edge label of the original resource's class: For each child connected to the original resource through the specified label: Invoke Extend to unify or add the child, recording the name of the resulting binding Change the links in the copy to point to the bindings of the children Return the name of the copy BindResource (bind_resource) Bind the first resource to the second resource Bind the second resource to the first resource

Detailed Description Text (52):
There are some aspects of the method that have not been explained, because they are related to the specific application of the graph algorithm to the resource tree. One point which benefits from further explanation is the following step in Bind. Do any resource-specific merging of the two resources

CLAIMS:

20. The method of claim 19, wherein descendants of said given node describe attributes of a resource.

L21: Entry 6 of 11 File: USPT Feb 5, 2002

DOCUMENT-IDENTIFIER: US 6345277 B1
TITLE: Method and apparatus for using an information model to organize an information repository into an extensible hierarchy of information

Application Filing Date (1):
19990226

Parent Case Text (1):
This application is related to the copending U.S. Application Ser. No. 09/258,576 of Jason Goldman and Brian O'Keefe entitled "METHOD AND APPARATUS FOR USING AN INFORMATION MODEL TO ORGANIZE AN INFORMATION REPOSITORY INTO A HIERARCHY OF INFORMATION", the copending U.S. Application Ser. No. 09/258,984 of Jason Goldman and D. Jon Lachelt entitled "METHOD AND APPARATUS FOR USING AN INFORMATION MODEL TO ORGANIZE AN INFORMATION REPOSITORY INTO AN EXTENSIBLE HIERARCHY OF ORGANIZATIONAL INFORMATION", and the copending U.S. Application Ser. No. 09/258,567 of Jason Goldman entitled "METHOD AND APPARATUS FOR USING AN INFORMATION MODEL TO CREATE A LOCATION TREE IN A HIERARCHY OF INFORMATION", and incorporates by reference all that is disclosed therein.

Detailed Description Text (15):
Throughout the drawing, value-defined container definition nodes in an information model are depicted as objects so that hierarchical relationships between value-defined container definition nodes can be illustrated. The hierarchical relationship used is the relationship illustrated in a preferred embodiment (FIG. 3). Derived containers are also depicted as objects similar in structure to their corresponding value-defined container definition nodes. In figures illustrating user-viewable elements, derived containers are shown only by their label. In these figures, a label can represent an actual derived container in a hierarchy and is the equivalent of a derived container object bearing the same label. These labels can also represent a derived container in a hierarchy structure, which is a blueprint from which a hierarchy is designed, and has the same characteristics of a hierarchy (i.e., it comprises derived containers, levels, etc.). Additionally, it is to be understood that all structures described herein (i.e. information models, hierarchies, information hierarchies, derived hierarchies, value-defined container definition nodes, derived containers, etc.) can have any number of conceivable representations (i.e. tree structures, list representations), as long as they are in conformance with the teachings herein.

Detailed Description Text (47):
Therefore, when derived containers 1916, 1938 are created, pointers 1924,1946, respectively, are created to reference corresponding type-defined container definition node 1900, which is an attribute-based container definition node since it generates more than one derived container. Descendant type-defined container definition nodes, if any exist, of a type-defined container definition node generate one set of derived containers for each derived container of its parent type-defined container definition node, where a set of derived containers comprises one derived container if a corresponding type-defined container definition node is value-based, or one or more derived containers if a corresponding type-defined container definition node is attribute-based. Therefore, since type-defined container definition node 1902 is a descendant of type-defined container definition

node 1900 that is attribute-based, it will generate a set of derived containers for each derived container of its parent. Since its parent 1900 is referenced by two derived containers 1916 and 1938, two derived containers are generated from type-defined container definition node 1902: derived container 1920 which comprises a pointer 1928 back to type-defined container definition node 1902, and derived container 1942 which comprises a pointer 1950 back to type-defined container definition node 1902. Type-defined container definition node 1904 is also a descendant of attribute-based container definition node 1900, and will generate derived container 1918 which comprises a pointer 1926 back to type-defined container definition node 1904, and derived container 1940 which comprises a pointer 1948 back to type-defined container definition node 1904. Finally, type-defined container definition node 1906 is also a descendant of attribute-based container definition node 1900, and will generate derived container 1922 which comprises a pointer 1930 back to type-defined container definition node 1906, and derived container 1944 which comprises a pointer 1952 back to type-defined container definition node 1906.

Detailed Description Text (55):
Since derived containers generated from the same attribute-based container definition node share the same parent/ancestors and children/descendants, each derived container generated from the same attribute-based container definition node should be hierarchically related to its parent/ancestors (top level) and its children/descendants (bottom level). Thus, the following should also be analyzed:

Detailed Description Text (74):
To create an extensible hierarchy in a second preferred embodiment, an information model is created by defining type-defined container definition nodes, so that designer intervention is not needed. Whereas in a first preferred embodiment one value-defined container definition node is defined for every derived container, in a second preferred embodiment, type-defined container definition nodes are strategically defined according to the desired hierarchy structure. Related derived containers that are also hierarchically related to their parent/lancestors and children/descendants in a hierarchy structure can be defined by a single attribute-based container definition node, and all other derived containers in the structure can be defined by a value-based container definition node.

CLAIMS:

13. Apparatus as in claim 11, wherein children and descendant derived containers of said related derived containers in said given group are determined by children and descendant type-defined container definition nodes of a parent attribute-based container definition node corresponding to said related derived containers in said given group.

First Hit Fwd Refs

☐ Generate Collection

L21: Entry 11 of 11 File: USPT Sep 19, 1989

DOCUMENT-IDENTIFIER: US 4868743 A
TITLE: Traversal method of processing tree structure information and apparatus using the same

Abstract Text (1):
A traversal method having a memory for storing tree structure data having nodes, each node including an attribute indicating a traversal stage in the tree structure in which an evaluation result of an attribute corresponding to the traversal state of each node of the tree structure data is stepwise arranged according to a depth-first order in the memory in a sequential fashion, the memory including two memories. In this method, the system recognizes as a sub-tree a group of nodes which have a same kind and which are linked with each other, sequentially arranges, during a traversal of a root of the sub-tree, symbols indicating an initiate position of the sub-tree and an end position thereof, and a symbol indicating an intermediate position of the sub-tree in a first memory. The system copies, while searching a series of data thus sequentially arranged for each kind, the initiate symbol, the intermediate symbol, and the end symbol of the sub-tree not belonging to the kind onto second memory. When the initiate symbol of a sub-tree belonging to the kind is detected, an activation of a traversal of the sub-tree is indicated, and when the intermediate and end symbols are detected, a reactivation of a traversal of the sub-tree is indicated. In response to the indication, a traversal of the sub-tree belonging to the kind is executed so as to develop an attribute of the sub-tree in the second memory.

Application Filing Date (1):
19871123

Brief Summary Text (2):
The present invention relates to a traversal method in processing tree structure information, and in particular, to a traversal method and an apparatus using the same, the method being capable of increasing an efficiency of a traversal of a tree when evaluation of information of nodes constituting the tree prior to or during a tree traversal is difficult.

Brief Summary Text (3):
A method in which data of a tree structure is subjected to a traversal or scanned to effect processing based on information of nodes of the tree structure according to the sequence of the traversal of the nodes has often been employed in various software including a compiler.

Brief Summary Text (4):
The object data of the scanning or traversal constitutes, for example, as shown in FIG. 15, a tree structure 200 having nodes 201 as components thereof and branches 202 connecting nodes. Furthermore, among the nodes 201 of the tree structure, the nodes at the upper-most position is called a root and a node which is not connected to a node therebelow is referred to as a leaf. In addition, for the relationship between nodes 201 connected to each other by a branch 202, a node y connected to a node x and is located immediately therebelow is called a child of the node x, which is in turn referred to as a parent. Moreover, for the relationship of nodes sharing a parent, children on the left-hand and right-hand sides of the parent node x are

respectively referred to as an elder child and a younger child, whereas a node on the right-hand side of the node x is named as a younger brother of the node x, which is accordingly referred to as the elder brother.

Brief Summary Text (5):
Incidentally, since these nodes configure a tree structure, each node retains the positions of data of the elder child and younger brother thereof. Furthermore, if there does not exist the elder child or the younger brother, a special value, nil is retained in place of the location thereof.

Brief Summary Text (6):
When effecting a traversal of a tree having such a structure, a depth-first-search is achieved, namely, starting from the root of the tree, the traversal is accomplished downward from the child at the left-most position. When all children of this sequence are completely searched, the similar search is executed from the child at the second left-most position. This operation is repeatedly effected toward the right-most location.

Brief Summary Text (7):
Furthermore, in a case where an operation is to be executed for each node during the search, there is used method in which the tree traversal is achieved in a sequence including all of a preorder in which the root is first processed for the operation for each node, an in order in which the root is processed at an intermediate point for the operation, and a post-order in which the root is finally processed, thereby sequentially arranging data according to the processing order.

Brief Summary Text (8):
For example, as shown in FIG. 16(a) a node R has three children. Applying the method above to the tree structure including sub-trees a-c to effect the tree traversal and to sequentially arrange data, there are sequentially attained in an arrangement a result R.sub.1 of the tree traversal (preorder) of the node R before the sub-trees a-c, a result R.sub.2 of the tree traversal (in order) of the node R between the sub-trees a and c, a result R.sub.3 of the tree traversal (in order) between the sub-trees b and c, and a result R.sub.4 of the tree traversal (postorder) of the node R after the sub-trees a-b.

Brief Summary Text (9):
As a consequence, in a tree traversal, the node assigned as an object of the tree traversal at present possesses states of which the number is expressed as the total of the children thereof +1. These states are referred to as traversal stages of the node and is represented by integers ranging from 0 to the total of the children thereof. Each node is provided with an attribute of each traversal stage, namely, contents R.sub.1 -R.sub.4 to be sequentially arranged or information indicating the contents R.sub.1 -R.sub.4.

Brief Summary Text (10):
The tree traversal method of this kind is particularly applied to cases, for example, a case where evaluation of the attributes of all nodes can be accomplished at a low cost prior to the start of the tree traversal or a case where although the evaluation of the respective attributes can be achieved according to the order of processing or visiting of nodes when the tree traversal is effected.

Brief Summary Text (11):
However, in a case where nodes are dependent on each other and hence the entire tree must undergo a tree traversal to evaluate attributes of a node or in a case where a great amount of information is required for the evaluation of the attribute and consequently the attributes of all nodes cannot be arranged through a tree traversal, the sequence of evaluation of attributes of each node is restricted. If the evaluation sequence does not agree with the tree traversal order on each node in the tree traversal, the respective operations cannot be achieved according to

WEST Search History

Hide Items | Restore | Clear | Cancel

DATE: Thursday, June 03, 2004

Hide?	Set Name	Query	Hit Count
		DB=PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD; PLUR=YES; OP=ADJ	
☐	L11	l9 and (mirror or mirroring)	6
☐	L10	L9 and l1	1
☐	L9	20010515	77
☐	L8	cache near8 (collect or gather) near8 (information or status)	106
☐	L7	20010515	7
☐	L6	(intelligent adj3 (cache or caching)) same (collaborate or collaboration or cooperate or cooperation)	8
☐	L5	L4 and l3	0
☐	L4	rolled adj2 up adj3 attribute	10
☐	L3	L2 and tree	54
☐	L2	20010515	226
☐	L1	intelligent adj3 (cache or caching)	337

END OF SEARCH HISTORY

WEST Search History

Hide Items | Restore | Clear | Cancel

DATE: Thursday, June 03, 2004

Hide?	Set Name	Query	Hit Count
		DB=PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD; PLUR=YES; OP=ADJ	
☐	L22	L21 and binary	0
☐	L21	L20 and tree	11
☐	L20	20010515	12
☐	L19	attribute near8 descendant near8 node	26
☐	L18	20010515	2
☐	L17	((initialize or initializing or configure or configuring) adj3 (node or device)) near8 (query or request) near8 (broadcast or multicast)	4
☐	L16	20010515	5
☐	L15	(content adj2(mirror or mirroring)).ti.	10
☐	L14	L13 and l1	1
☐	L13	20010515	62685
☐	L12	(mirror or mirroring).ti.	82289
☐	L11	l9 and (mirror or mirroring)	6
☐	L10	L9 and l1	1
☐	L9	20010515	77
☐	L8	cache near8 (collect or gather) near8 (information or status)	106
☐	L7	20010515	7
☐	L6	(intelligent adj3 (cache or caching)) same (collaborate or collaboration or cooperate or cooperation)	8
☐	L5	L4 and l3	0
☐	L4	rolled adj2 up adj3 attribute	10
☐	L3	L2 and tree	54
☐	L2	20010515	226
☐	L1	intelligent adj3 (cache or caching)	337

END OF SEARCH HISTORY